Direction-of-Change Financial Time Series Forecasting using Bayesian Learning for MLPs

Andrew A. Skabar

Abstract—**Conventional neural network training methods find a single set of values for network weights by minimizing an error function using some gradient descent-based technique. In contrast, the Bayesian approach infers the posterior distribution of weights, and makes predictions by averaging the predictions over a sample of networks, weighted by the posterior probability of the network given the data. The integrative nature of the Bayesian approach allows it to avoid many of the difficulties inherent in conventional approaches. This paper reports on the application of Bayesian MLP techniques to the problem of predicting the direction in the movement of the daily close value of the Australian All Ordinaries financial index. Predictions made over a 13 year out-of-sample period were tested against the null hypothesis that the mean accuracy of the model is no greater than the mean accuracy of a coin-flip procedure biased to take into account non-stationarity in the data. Results show that the null hypothesis can be rejected at the 0.005 level, and that the t-test p-values obtained using the Bayesian approach are smaller than those obtained using conventional MLPs methods.**

Index Terms—**Direction-of-change forecasting, Financial time series, Markov Chain Monte Carlo, Neural Networks.**

I. Introduction

Predicting the future value of a time series based on historical values is usually approached as an auto-regression problem in which the parameters of the model are first optimized using in-sample data; the model is then used to make forecasts on a set of out-of-sample data; and the accuracy of the forecasts is finally measured by comparing the forecast values with the realized (i.e., actual) values. In many cases, however, correctly predicting the direction of the change (i.e., *up* or *down*) is a more important measure of success. For example, if a trader is to make buy/sell decisions on the basis of forecast values, it is more important to be able to correctly predict the direction of change than it is to achieve, say, a small mean squared error. We call this *direction-of-change forecasting*, and its importance has been acknowledged in several recent papers [1]-[4].

Clearly, if one has made a prediction for the future value of a time-series, then that prediction can trivially be converted into a direction-of-change prediction by simply predicting up/down if the forecast value is greater/smaller than the present value. There may, however, be advantages in predicting the direction-of-change directly (i.e., without explicitly predicting the future *value* of the series). For example, traders base their decisions primarily on their opinion of whether the price of a commodity will rise or fall, and to a lesser extent on their opinion of the degree to which it will rise or fall. This may create in financial systems an underlying dynamic that allows the direction of change to be predicted more reliably than the value of the series. In this paper, we perform direction-of-change forecasting using Multilayer Perceptrons (MLPs) as binary classifiers.

Conventional MLP training methods attempt to find a single set of values for the network weights by minimizing an error function using some gradient descent based technique. The error function is usually chosen such that the resulting network represents the most probable network given the data, and following Bishop (1995), we refer to this as the *maximum likelihood* (ML) approach [5]. In order to prevent overfitting, and thus achieve good predictive performance on holdout data, the network should have an appropriate number of hidden layer units, and the error function should usually include a regularization term. Model parameters such the number of hidden units and the value of the regularization coefficients should be determined before the final network is trained, and a validation procedure is usually used to determine these values.

In contrast to the ML approach, Bayesian MLP methods [6-8] do not attempt to find a single 'best' weight vector; but rather, they attempt to infer the posterior distribution of the weights, given the data. Samples can then be taken from this distribution, each sample representing a distinct MLP. Given some novel example, each of the sampled networks can be applied to the example, with the resulting prediction being the average prediction over the sample of networks weighted by the posterior probability of the network given the training data. Thus, whereas the conventional approach *optimizes* over parameters, the Bayesian approach *integrates* over parameters, and this allows it to avoid many of the difficulties that conventional approaches have in avoiding overfitting.

This paper reports on the application of Bayesian MLP methods to the financial prediction problem, and expands on preliminary results that have been presented in [9]. Section 2 describes the neural network approach to the problem of financial prediction, and highlights some of the problems that conventional MLP approaches have in this domain. Section 3 describes the Bayesian MLP approach, and Section 4 presents empirical results of applying Bayesian MLP techniques to predicting the direction of change in daily close value of the Australian All Ordinaries index. The results are discussed in Section 5, and Section 6 concludes the paper.

Manuscript received March 13, 2008. A. Skabar is with the Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Australia. (phone: (+61) 3 9479 2598; fax: (+61) 3 9479 3060; e-mail: a.skabar@latrobe.edu.au).

II. MLPs for Financial Prediction

A *multilayer perceptron* (MLP) is a function of the following form:

$$f(\mathbf{x}^n) = h(u) \text{ where } u = \sum_{j=0}^{N_1} w_{kj}\, g\left(\sum_{i=0}^{N_0} w_{ji} x_i^n\right) \quad (1)$$

where N_0 is the number of inputs (i.e., the dimensionality of the input feature vector), N_1 is the number of units in a hidden layer, w_{ji} is a numerical weight connecting input unit i with hidden unit j, w_{kj} is the weight connecting hidden unit j with output unit k, $h(u) = \sigma(u) \equiv (1 + \exp(-u))^{-1}$ (i.e., a *sigmoid* function), and $g(u)$ is either a sigmoid, or some other continuous, differentiable, nonlinear function. Thus, an MLP with some given architecture, and weight vector **w**, provides a mapping from an input vector **x** to a predicted output y given by y = f (**x**, **w**). Given some data, D, consisting of n independent items ($\mathbf{x}^1$, y^1), …, ($\mathbf{x}^N$, y^N), the objective is to find a suitable **w**.

In the case of financial prediction, the raw data usually consists of a series of values measured at regular time intervals; *e.g.* the daily close value of a financial index such as the Australian All Ordinaries. The input vector, $\mathbf{x}^N$, corresponds to the N^{th} value of the series, and usually consists of the current value, in addition to time lagged values, or quantities which can be derived from the series, such as moving averages.

One approach is to use the MLP to predict the next value of the series. This is a regression problem, as the objective is to predict a continuous-valued quantity; *i.e.*, the price of a stock, or the close value of an index. In this case, the appropriate error function to be minimized is the quadratic error between target and predicted outputs. This is justified under the assumption that the training examples are normally distributed around the target function with zero mean and constant variance.

The other approach, and the approach taken in this paper, is to predict only the direction of the change. In order to do this, one could simply take the regression approach described above, and compare the predicted (continuous) value against the current value to determine the direction of change. However, an alternative approach is to treat the problem as a binary classification problem. In this case, the training examples are labelled with binary target values representing the direction of change from the previous day's value. In order for the MLP to represent the *probability* of an upwards change (target value of 1), the MLP should be trained so as to minimize the *cross-entropy error*, and is described in greater detail in the next section.

The conventional approach to finding the weight vector **w** is to use a gradient-descent method to find a weight vector that minimizes the error between the network output value, f (**x**, **w**), and the target value, y. This approach is generally referred to as the *maximum likelihood* (ML) approach because it attempts to find the most probable weight vector, given the training data. This weight vector, $\mathbf{w}_{\mathbf{ML}}$, is then used to predict the output value corresponding to some new input vector $\mathbf{x}^{n+1}$.

One of the main difficulties in applying MLPs to the financial prediction domain concerns the very high level of noise in the data, and thus the danger of overfitting the network. The main issue is how to optimize model parameters, such as the number of hidden units and regularization coefficient, so as to minimize the degree of overfitting on the training data. One approach is to use an independent validation set to optimize these parameters. The main difficulty here is how to select examples for the validation set. For example, if these are chosen to be adjacent to, but between, the training and test sets, then any patterns found in the training data may have dissipated before the model is applied to the test data. Moreover, because the validation set is itself noisy, there will be considerable uncertainty in whether the parameters values chosen are indeed optimal. An alternative approach is to omit the validation set, and select parameters that provide the best results on the test data, but in this case we can never be sure that we have not simply optimized these parameters to the test set.

III. Bayesian Methods for MLPs

In contrast to the ML approach, Bayesian methods for MLPs do not attempt to find a single 'best' weight vector **w**; rather, they infer $p(\mathbf{w}|D)$, the posterior distribution of the weights given the data. The predicted output corresponding to some input vector $\mathbf{x}^n$ is then obtained by performing a weighted sum of the predictions over all possible weight vectors, where the weighting coefficient for a particular weight vector depends on $p(\mathbf{w}|D)$. Thus,

$$\hat{y}^n = \int f(\mathbf{x}^n, \mathbf{w})\, p(\mathbf{w} \mid D)\, d\mathbf{w} \quad (2)$$

where $f(\mathbf{x}^n, \mathbf{w})$ is the MLP output. The fact that $p(\mathbf{w}|D)$ is a probability density function allows us to express the integral in Equation 2 as the expected value of $f(\mathbf{x}^n, \mathbf{w})$ over this density:

$$\begin{aligned} \int f(\mathbf{x}^n, \mathbf{w}) p(\mathbf{w} \mid D)\, d\mathbf{w} &= E_{p(\mathbf{w}|D)}\left[f(\mathbf{x}^n, \mathbf{w}) \right] \\ &\simeq \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}^n, \mathbf{w}) \end{aligned} \quad (3)$$

Thus, the integral can be estimated by drawing N samples from the density $p(\mathbf{w}|D)$, and averaging the predictions due to these samples. This process is known as *Monte Carlo* integration.

The density $p(\mathbf{w}|D)$ can be estimated using the fact that $p(\mathbf{w} \mid D) \propto p(D \mid \mathbf{w}) p(\mathbf{w})$, where $p(\mathbf{w}|D)$ is the likelihood, and $p(\mathbf{w})$ is the prior weight distribution. If the target values are binary, then the likelihood can be expressed as

$$p(D \mid \mathbf{w}) = \exp \sum_n \left\{ t^n \ln f(\mathbf{x}^n, \mathbf{w}) + (1-t)^n \ln(1 - f(\mathbf{x}^n, \mathbf{w})) \right\} \quad (4)$$

where, for the financial prediction problem, $t^n = 1$ if the close value for day n+1 is greater than that for day n (*i.e.*, an upwards movement) and 0 otherwise (downwards movement).

The prior weight distribution, $p(\mathbf{w})$, should reflect any prior knowledge that we have about the complexity of the MLP. To reflect the fact that we want it to be a smooth function, $p(\mathbf{w})$ is commonly assumed to be Gaussian with zero mean and inverse variance α, giving preference to weights with smaller magnitudes; *i.e.*,

$$p(\mathbf{w}) = \left(\frac{\alpha}{2\pi}\right)^{m/2} \exp\left(-\frac{\alpha}{2}\sum_{i=1}^{m} {w_i}^2\right) \tag{5}$$

where m is the number of weights in the network [12]. However, we usually do not know what variance to assume for the prior distribution, and for this reason it is common to set a distribution of values. As α must be positive, a suitable form for its distribution is the gamma distribution [12]. Thus,

$$p(\alpha) = \frac{(a/2\mu)^{a/2}}{\Gamma(a/2)} \alpha^{a/2-1} \exp(-\alpha a/2\mu) \tag{6}$$

where the a and μ are respectively the shape and mean of the gamma distribution, and are set manually. Note that a single α need not be used for all weights and biases. For example, it is common to use separate values of α for input-hidden-layer weights, input-to-hidden-layer biases, hidden-to-output layer weights, and hidden-to-output layer biases, which is the approach taken in this paper, and in which we denote the respective weight groupings as α_1, α_2, α_3, and α_4 respectively. Note that each weight grouping will have its own distribution parameterised by a_i and μ_i, where a_i and μ_i represent the shape and mean for the respective group. We discuss methods for selecting appropriate values for a and μ in the next section.

Because the prior depends on α, Equation 2 should be modified such that it includes the posterior distribution over the α parameters:

$$\hat{y}^n = \int f(\mathbf{x}^n, \mathbf{w})\, p(\mathbf{w}, \alpha \mid D)\, d\mathbf{w} d\alpha \tag{7}$$

where

$$p(\mathbf{w}, \alpha \mid D) \propto p(D \mid \mathbf{w})\, p(\mathbf{w}, \alpha) \tag{8}$$

Monte Carlo integration depends on the ability to obtain samples from the posterior distribution. The objective is to sample preferentially from the region where $p(\mathbf{w},\alpha \mid D)$ is large. The Metropolis algorithm [10] achieves this by generating a sequence of vectors in such a way that each successive vector depends on the previous vector as well as having a random component; i.e., $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \varepsilon$, where ε is a small random vector. Preferential sampling is then achieved using the criterion:

if $p(\mathbf{w}_{\text{new}} \mid D) > p(\mathbf{w}_{\text{old}} \mid D)$ accept

if $p(\mathbf{w}_{\text{new}} \mid D) < p(\mathbf{w}_{\text{old}} \mid D)$ accept with probability $\dfrac{p(\mathbf{w}_{\text{new}} \mid D)}{p(\mathbf{w}_{\text{old}} \mid D)}$

The difficulty in using the Metropolis algorithm to estimate the integrals for neural networks stems from the strong correlations in the posterior weight distribution; *i.e.*, the great majority of the candidate steps generated in the random walk will be rejected as they lead to a decrease in $p(\mathbf{w}|D)$ [5]. The Hybrid Monte Carlo algorithm [11] reduces the random walk behaviour by using gradient information, which, in the case of MLPs, can be readily calculated. While the Hybrid Monte Carlo algorithm allows for the efficient sampling of parameters (*i.e.,* weights and biases), the posterior distribution for α should also be determined. In this paper we use Neal's (1996) approach, and use Gibbs sampling [12] for the αs.

IV. Empirical Results

The experimental results reported in this section were obtained by applying the techniques described previously to the daily close values of the Australian All Ordinaries Index (AORD) for the period from November 1990 to December 2004. The task is to predict the direction of movement (*up* or *down*) of the close value on day t+1 from historical close values. Specifically, we are interested in the proportion of test examples for which the direction in movement is predicted correctly, and we refer to this as the *sign correctness proportion* (SCP). The SCP is defined as follows:

$$SCP = \frac{1}{N}\sum_{k=1}^{N} \delta_{t_k o_k} \tag{9}$$

where N is the number of test examples in the prediction period, t_k and o_k are the binary target and network outputs respectively, and δ_{ij} = 1 if $i = j$, and 0 otherwise. Note that because the network output represents the probability of an upward movement, the binary output, o_k, is obtained by thresholding the output at 0.5.

The time series is shown in Figure 1, in which days 0 and above are those for which prediction were made, and days prior to day 0 were used for preprocessing. In total, 3600 predictions were made. Note that the series is clearly non-stationary.

A. Feature Selection and Preprocessing

Almost invariably, successful financial forecasting requires that some preprocessing be performed on the raw data. This usually involves some type of transformation of the data into

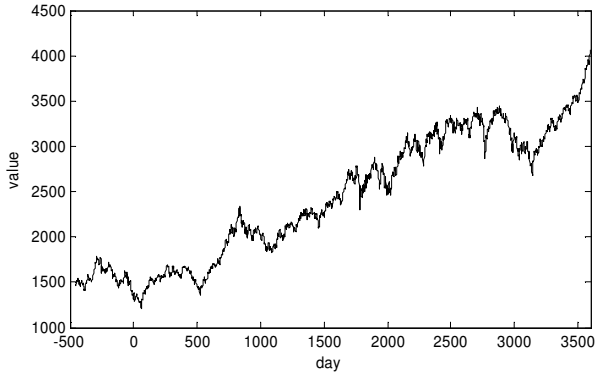

Fig. 1. Daily close values of Australian All Ordinaries Index (AORD) from January1989 to December 2004. Prediction period begins from day 0.

new variables more amenable to learning from. Thus, rather than using, for example, the past prices of a stock, transformed variables might include the absolute change in the price, $(p_t - p_{t-1})$, the rate of change of price, $(p_t - p_{t-1}) / p_{t-1}$, or the price or change in price relative to some index or other baseline such as a moving average [3]. The advantage of using variables based on changes in price (either relative or absolute) is that they help to remove non-stationarity from the time series.

In this study, the input variables used are the relative change in close value from the previous day to the current day (r_1), and the 5, 10, 30 and 60 day moving averages (ma_5, ma_{10}, ma_{30}, ma_{60}). The moving averages are calculated by simply averaging the x previous closing values, where x is the period of the moving average. Thus, the input to the network is the vector ($r_1(t)$, $ma_5(t)$, $ma_{10}(t)$, $ma_{30}(t)$, $ma_{60}(t)$) where

$$r_n(t) = \left(p_t - p_{t-n} \right) / p_t \tag{10}$$

and

$$ma_n(t) = \frac{1}{n} \sum_{t-n}^{t} p_t \tag{11}$$

Note that there would almost certainly exist some other combination of inputs and preprocessing steps that might lead to better performance than that which can be achieved using this particular combination. However, in this paper we are not concerned with optimizing this choice of inputs; rather, we are concerned with comparing the performance of the Bayesian approach with that of the ML approach. If the Bayesian approach is found to give superior performance, this is very unlikely to be due to this particular choice of inputs.

B. Training and Prediction Windows

A prediction window period of 30 days was used in this study, with the training set for each 30-day prediction period consisting of data for the 200 trading days immediately preceding the test period. A total of 120 30-day predictions were made (*i.e.*, 3600 days), with the training and test windows advanced by 30 days after each 30-day prediction period. This is depicted in Figure 2.

The choice of 200 days for training was based on the assumption that patterns that exist in the training may dissipate after some time, and hence that data temporally far removed from the prediction period may not be useful. In principle, the test window could consist of a single prediction, but this would increase the computational cost 30-fold. Also, the particular hypothesis testing scheme that we use requires that the test window consists of multiple predictions (see below).

C. Hypothesis Testing

Assuming that a return series is stationary, then a coin-flip decision procedure for predicting the direction of change would be expected to result in 50% of the predictions being correct. We would like to know whether our model can produce predictions which are statistically better than 50%.

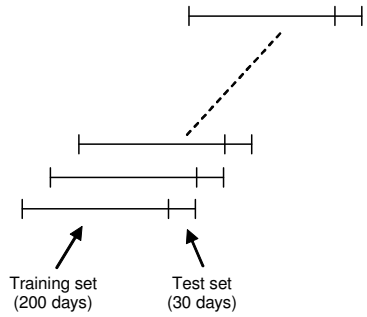

Fig. 2. Training and test windows. Windows are advanced by 30 days after each 30-day prediction is made

However, a problem is that many financial return series are not stationary, as evidenced by the tendency for commodity prices to rise over the long term. This is clearly visible in Fig 1, where there is a clear upward trend over the period shown, suggesting that the total number of upwards movements is greater than the number of downward movements.Thus it may be possible to achieve an accuracy significantly better than 50% by simply biasing the model to always predict up.

A better approach is to compensate for this non-stationarity, and this can be done as follows. Let x_a represent the fraction of days in an out-of-sample test period for which the *actual* movement is up, and let x_p represent the fraction of days in the test period for which the *predicted* movement is up. Therefore under a coin-flip model the expected fraction of days corresponding to a correct upward prediction is $(x_a \times x_p)$, and the expected fraction of days corresponding to a correct downward prediction is $(1-x_a) \times (1-x_p)$. Thus the expected fraction of correct predictions is

$$a_{exp} = (x_a \times x_p) + ((1-x_a) \times (1-x_p)) \tag{12}$$

We wish to test whether a_{mod} (the accuracy of the predictions of our model) is significantly greater than a_{exp} (the compensated coin-flip accuracy). Thus, our null hypothesis may be expressed as follows:

$$\textit{Null Hypothesis}: \; H_0: \; a_{mod} \le a_{exp} \quad H_1: \; a_{mod} > a_{exp}$$

The null hypothesis can be tested by performing a paired t-test of the samples obtained from each of the 120 30-day prediction periods, comparing the means of a_{mod} and a_{exp}.

D. Setting the Prior Distribution

The Bayesian approach requires that we specify the prior weight distribution, $p(\mathbf{w})$. Recall that $p(\mathbf{w})$ is assumed to be Gaussian, with inverse variance α, and that α is assumed to be distributed according to a Gamma distribution with shape a and mean μ., which remain to be specified. In order to gain some insight into the range of values may be suitable for these parameters, we conducted a number of trials using the ML approach, with weight optimization performed using the scaled conjugate gradients algorithm.

Table I shows the training and test accuracies corresponding to various values of α. Accuracies are averaged over the 120 30-day prediction windows. The values in the fourth column of the table represent the p-values

obtained through performing the t-test. Italics show best performance.

Figure 3 plots training and test set accuracies against the α value. Low values for α, such as 0.01, impose a small penalty for large weights, and result in overfitting; *i.e.,* high accuracy on training data, but low accuracy on test data. In this case, the null hypothesis cannot be rejected at the 0.05 level. In contrast, when α is very high (*e.g.*, 10.0), large weights will penalised more heavily, leading to weights with small magnitudes. In this case the MLP will be operating in its linear region and the MLP will display a strong bias towards predictions that are in the same direction as the direction of the majority of changes on the training data. Thus, if the number of upward movements on the training data is greater than the number of negative movements, the MLP will be biased towards making upwards predictions on the test data; however, this is not likely to lead to a rejection of the null hypothesis because the null hypothesis takes the non-stationarity of the data into account. It can be seen from Figure 3 that a local maximum for the test set accuracy occurs for an α value of 1.0, and in this case the null hypothesis can clearly be rejected at the 0.01 level.

The range of α values for which the null hypothesis can be rejected is very narrow, and ranges from a lower α value in the vicinity of 0.5 to 0.75, to a an upper α value in the vicinity of 1.5 to 2.0. After visualizing the pdf for Gamma distributions with mean 1.0 and various values for the shape parameter, a shape parameter of 10 was chosen. The pdf is shown in Figure 4. Note that the pdf conforms roughly to the α value identified in the Table I as leading to a rejection of the null hypothesis.

TABLE I.
TRAIN ACCURACY, TEST ACCURACY AND P-VALUE FOR VARIOUS α VALUES

α value	*Train. Acc.*	*Test. Acc.*	*p-value (Ho)*
0.010	0.725	0.490	0.402
0.100	0.701	0.501	0.459
0.500	0.608	0.516	0.169
0.750	0.593	0.520	0.070
1.000	***0.585***	***0.524***	***0.007***
1.500	0.570	0.521	0.038
2.000	0.562	0.518	0.587
5.000	0.549	0.526	0.528
10.00	0.542	0.525	0.479

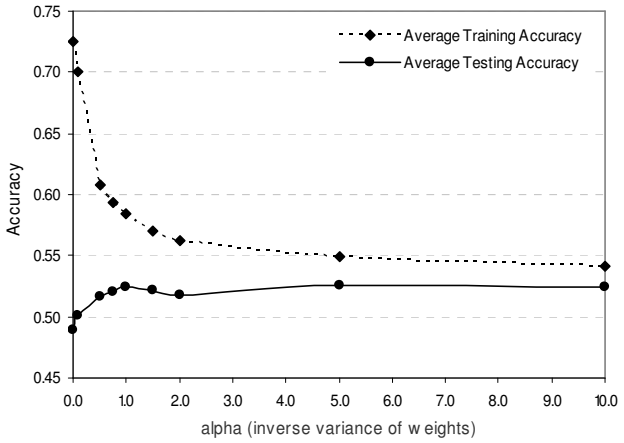

Fig. 3. Training and test set accuracy averaged over 120 training/test set pairs. A local maximum test accuracy corresponds to an α value of approximately 1.0

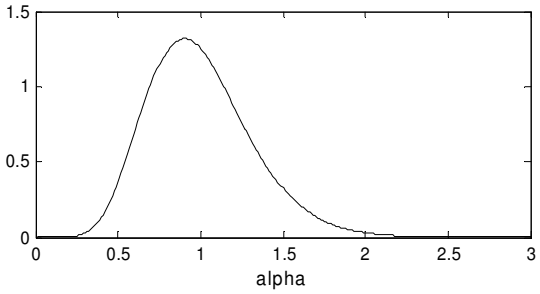

Fig. 4. Gamma distribution with mean 1.0 and shape parameter value 10

E. MCMC sampling

We now describe the application of the Bayesian approach, which relies on MCMC sampling to draw weight vectors from the posterior weight distribution.

Monte Carlo sampling must be allowed to proceed for some time before the sampling converges to the posterior distribution. This is called the *burn-in period*. In the results presented below, we allowed a burn-in period of 1000 samples, following which we then saved every tenth sample until a set of 100 samples was obtained. Each of the 100 samples was then applied to predicting the probability of an upwards change in the value of the index on the test examples, and the probabilities were then averaged over the 100 samples. The resulting *p*-values are shown in Table II, together with the results from Table I corresponding to an α value of 1.0; *i.e.*, the performance of the best network trained using the conventional approach. Note that the p-values are now much smaller, indicating increased confidence in the rejection of the null hypotheses, and that the average test accuracy has increased from 52.4% to 52.8%. Also note that the average training accuracy for the Bayesian approach is less than that for the ML approach, thereby supporting the claim that the Bayesian approach is better at avoiding overfitting.

V. DISCUSSION

The superior performance of the Bayesian approach can be attributed to its integrative nature: each individual weight vector has its own bias, and by integrating over many weight vectors, this bias is decreased, thus reducing the likelihood of inferior generalization performance resulting from overfitting on the training data.

The Bayesian technique assumes that the MCMC sampling has proceeded for a sufficient time such that the convergence to the true posterior distribution has been achieved; however, there are currently no adequate techniques available for testing convergence. We experimented with starting the MCMC sampling from various starting weight vectors. Specifically, we compared the results of starting the sampling

TABLE II
TRAIN ACCURACY, TEST ACCURACY AND P-VALUE FOR CONVENTIONALLY TRAINED (SCG) AND BAYESIAN-TRAINED (MCMC) MLPS

Method	*Train. Acc.*	*Test. Acc.*	*p-value (Ho)*
SCG	0.585	0.524	0.0068
MCMC	0.571	0.528	0.0011

from a random point, with results of sampling from a minima found by applying the ML approach, and found that after the 1000 sample burn-in period there was no significant difference in results. While this is not conclusive evidence that the sampling has converged, it does indicate that the 1000 sample burn-in period is sufficient to bring the sampling into the region of weight space corresponding to a local minimum, and probably close to the global minimum.

The most important decision to be made in using the Bayesian approach is the choice of prior. In this study, we used a relatively narrow distribution for α, the parameter controlling the degree of regularization present in the error function. This choice was made based on experimenting with different α values within the ML approach. The criticism could be made that this prior distribution was selected based on the same data that we had used for testing, and hence that the significance of our results may be overstated; however, this is unlikely to be the case, as the prior depends on factors such as the degree of noise in the data, and this is relatively constant over different periods of the same index. Consequently, it is very unlikely that using a different set of data would have resulted in a significantly prior. Moreover, the fact that we need only select parameters describing the *distribution* of α, rather than specific value for α, further diminishes the possibility that our prior is biased towards the particular dataset that we have used.

One of the advantages of the Bayesian approach is its inherent ability to avoid overfitting, even when using very complex models. Thus, although the results presented in this paper were based on MLPs with six hidden units, the same performance could, in principle, have been achieved using a more complex network. It is not clear, however, whether we should expect any coupling between the number of hidden layer units and the prior distribution. For this reason, we would recommend preliminary analysis using the ML approach to ascertain the appropriate range for α and selection of priors based on values which lead to significant predictive ability.

VI. Conclusion

Bayesian Learning for MLPs has been applied to predicting the next day's close value of the Australian All Ordinaries index over a period of approximately 13 years. Predictions were tested against the null hypothesis that the mean accuracy of the model is no greater than the mean accuracy of a coin-flip procedure biased to take into account non-stationarity in the data. Results show that the null hypothesis can be rejected at the 0.005 level, and that the t-test p-values obtained using the Bayesian approach are approximately one fifth the value of those obtained using MLPs trained using the conventional ML approach. The superior performance of the Bayesian approach is due to its integrative nature, and its inherent ability to avoid overfitting.

References

[1] Y. Kajitani, A.I. Mcleod, K.W. Hipel, "Forecasting nonlinear time series with feed-forward neural networks: a case study of Canadian lynx data," *Journal of Forecasting* 24, 2005, pp. 105–117,

[2] J. Chung, Y. Hong, "Model-free evaluation of directional predictability in foreign exchange markets,".*Journal of Applied Econometrics* 22, 2007, pp. 855–889.

[3] P.F. Christoffersen, F.X. Diebold, "Financial asset returns, direction-of-change forecasting, and volatility dynamics," PIER Working Paper Archive 04-009, Penn Institute for Economic Research, Department of Economics, University of Pennsylvania, 2003.

[4] S. Walczak, "An empirical analysis of data requirements for financial forecasting with neural networks," *Journal of Management Information Systems*, 17 (4), 2001, pp. 203-222.

[5] C. Bishop, *Neural networks for pattern recognition*, Oxford University Press, Oxford, 1995.

[6] D.J.C. MacKay, "A practical Bayesian framework for back propagation networks," *Neural Computation*, 4(3), 1992, pp. 448-472.

[7] R.M. Neal, "Bayesian Training of Backpropagation Networks by the Hybrid Monte Carlo Method," Technical Report CRG-TR-92-1, Department of Computer Science, University of Toronto, 1992.

[8] R.M. Neal, *Bayesian Learning for Neural Networks*, New York: Springer-Verlag, 1996.

[9] A. Skabar, "Application of Bayesian Techniques for MLPs to Financial Time Series Forecasting," *Proceedings of 16th Australian Conference on Artificial Intelligence (AI 2005)*, 2005, pp. 888-891.

[10] N.A. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics* 21(6), 1953, pp. 1087-1092.

[11] S. Duane, A.D. Kennedy, B.J. Pendleton and D. Roweth, "Hybrid Monte Carlo". *Physics Letters B.* 195(2), 1987, pp. 216-222.

[12] S. Geman and G. Geman, G, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, 1984, pp. 721-741.